Geração de Testes de Desempenho e Estresse a partir de Testes Funcionais

Ismayle de Sousa Santos[1]
Pedro de Alcântara dos Santos Neto[1]
Rodolfo Sérgio Ferreira Resende[2]

Resumo: Apesar do teste de software estar relacionado à garantia de qualidade, ele demanda tempo e recursos, por isso acaba por não ser realizado em parte dos projetos executados. No contexto das empresas que sempre realizam testes, percebe-se que o teste funcional cresce em utilização e que o teste de desempenho e estresse ainda é pouco executado, mesmo diante do aumento do número de sistemas acessíveis via Web, nos quais esses testes são considerados essenciais. Neste artigo é proposto um método e sua ferramenta de apoio à geração de testes de desempenho e estresse de sistemas Web a partir de testes funcionais, na tentativa de popularizar o teste para esses requisitos não-funcionais. Apresenta-se também um estudo experimental que concluiu que o uso do método e da ferramenta pode gerar ganhos em termo de redução de esforço na criação desses testes de requisitos não-funcionais.

Abstract: Despite the fact that the software testing is related to quality assurance, in order to save time and resources, it is not performed in part of software projects. In the organizations where testing is performed it can be seen that functional testing is getting more common and that the stress and

[1] UFPI/DIE, Teresina-PI, {ismayle,pasn}@ufpi.edu.br
[2] UFMG/DCC, Belo Horizonte-MG, rodolfo@dcc.ufmg.br

performance tests are scarcely performed despite the increment of web-based systems where the latter are considered essential. In this article it is proposed a method and its supporting tool for stress and performance test generation of web based systems, in order to facilitate the execution of non-functional testing. The stress and performance test generation takes advantage of functional tests. Experimental studies are also presented showing that the method and the tool can generate overall effort reduction for these tests generation.

1 Introdução

Em um contexto de alta competitividade como é o atual, principalmente quando se fala em sistemas *Web*, é importante que as expectativas do cliente sejam atendidas o mais plenamente possível, caso contrário, o sistema desenvolvido será rapidamente sobrepujado pelos concorrentes [1]. Essas expectativas podem estar explícitas, quando o cliente exige que o sistema desempenhe determinada função, ou podem ser necessidades implícitas, tais como a necessidade que o sistema seja facilmente utilizável ou que responda às requisições em um tempo máximo [2].

A atividade de teste é parte importante do processo de garantir que essas expectativas sejam atendidas. Os testes, entretanto, requerem tempo, conhecimento, planejamento, infra-estrutura e pessoal especializado sendo, portanto, uma atividade inerentemente custosa [1]. Esses são os principais motivos que fazem com que algumas empresas desenvolvedoras de *software* não realizem testes. No entanto, a ausência dessa atividade pode deixar passar falhas que podem causar prejuízos irreparáveis tanto para a empresa desenvolvedora quanto para o próprio cliente. Um exemplo disso aconteceu no Serviço de Emergência de Londres em 1992 [3]. Nesse episódio houve perda de vidas devido a um aumento nas solicitações e incapacidade do sistema em tratar um grande número de requisições simultâneas. Testes de desempenho e estresse poderiam ter detectado esse problema, evitando assim a catástrofe.

O cenário atual relacionado à execução de testes de *software* pode ser resumido da seguinte forma: boa parte das organizações realiza o teste funcional, uma vez que ele envolve conceitos mais difundidos; mas por outro lado, poucas organizações realizam os testes de desempenho e estresse, embora seja notoriamente reconhecida sua importância, principalmente quando se trata de sistema para *Web*. Isso pode ser verificado nos dados obtidos do Programa Brasileiro da Qualidade e Produtividade de *Software* [4], que indicam que 54% das organizações executam testes de aceitação (que são basicamente testes funcionais), porém não há qualquer referência a testes de desempenho e estresse.

A partir desse cenário, pode-se observar que uma contribuição significativa para as organizações desenvolvedoras de *software* seria criar, de alguma forma, um mecanismo que auxiliasse o desenvolvimento de testes de desempenho e estresse a partir de algum artefato gerado durante o processo de desenvolvimento. Como as empresas que têm a intenção de realizar testes de desempenho e estresse já executam, em sua grande maioria, testes funcionais, esses últimos se tornam, portanto, ótimos candidatos para serem usados na geração dos testes de desempenho e estresse. Além disso, os testes funcionais possuem boa parte das informações necessárias, que são as funcionalidades a serem exercitadas, os dados a serem utilizados e os resultados esperados, o que garante a viabilidade da idéia proposta. Assim, apresenta-se neste artigo uma abordagem inovadora: a geração de testes de desempenho e estresse a partir de testes funcionais. Essa geração tem como objetivo reduzir os custos e o tempo gasto com esses testes, a partir da utilização dos testes funcionais, já consolidados em grande parte das organizações desenvolvedoras de *software*.

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta, com maiores detalhes, as diferenças entre testes funcionais, de desempenho e de estresse; a Seção 3 descreve o método para geração de testes de desempenho e estresse a partir de testes funcionais; a Seção 4 apresenta a ferramenta protótipo construída; a Seção 5 exibe os resultados de um estudo experimental realizado com o intuito de avaliar o método e a ferramenta; a Seção 6 apresenta os principais trabalhos relacionados; e a Seção 7, apresenta uma discussão final e direções para trabalhos futuros.

2 Testes Funcionais x Testes de Desempenho x Testes de Estresse

O teste de *software* pode ser categorizado de acordo com os objetivos da atividade. O teste funcional, por exemplo, tem como objetivo verificar o comportamento do *software*. Isso normalmente é feito a partir de uma análise da execução, verificando se a utilização de determinadas entradas causa a obtenção da saída esperada. A partir do teste funcional é possível validar o comportamento, de maneira a identificar possíveis incorreções, como por exemplo, ausência de certas regras de negócio. O teste funcional tende a ser mais simples e rápido de se executar, quando comparado a testes com outros objetivos, uma vez que a verificação e validação em um nível mais baixo costumam ser mais demoradas que no nível de interface com o usuário. Em virtude disso, esse é o teste mais conhecido e utilizado nas empresas desenvolvedoras de *software* [4].

A Figura 1 apresenta um teste funcional desenvolvido utilizando a ferramenta Selenium[3]. Em geral, as ferramentas para testes funcionais gravam as ações executadas e as reproduzem sempre que solicitado. No caso do Selenium e conforme exibido na Figura 1, existem comandos para acessar URLs (*open*), entrar com dados em campos (*type*), acionar comandos (*clickAndWait*) e verificar resultados (*assertText*). Na figura são exibidos três testes, todos relacionados ao teste do *login* em uma aplicação. A partir deles, foi verificado

[3] http://seleniumhq.org/

se a tentativa de *login* com um usuário válido, porém com senha incorreta (“ti”, “123456”) causava a exibição de uma mensagem de erro (“Não foi possível realizar o *login*.”), assim como a tentativa de *login* com um usuário inexistente (“errado”, “123456”). Por último, foi testado se o *login* com a senha correta (“ti”, “Java”) resultava no surgimento dos elos para as funcionalidades “Cadastro”, “Relatórios” e “Sair” (linhas 10 a 15).

O teste de desempenho consiste em verificar se um sistema atende aos seus requisitos de desempenho em um determinado contexto [5]. O contexto é fundamental para o teste e define o ambiente no qual as verificações devem ser realizadas [6]. Um exemplo de um contexto bem definido seria dizer: o sistema deve suportar 100 usuários simultâneos, com tempo de resposta máximo de 8 segundos. A partir dessa definição, o teste de desempenho deve simular o ambiente desejado e verificar se o sistema se comporta conforme especificado.

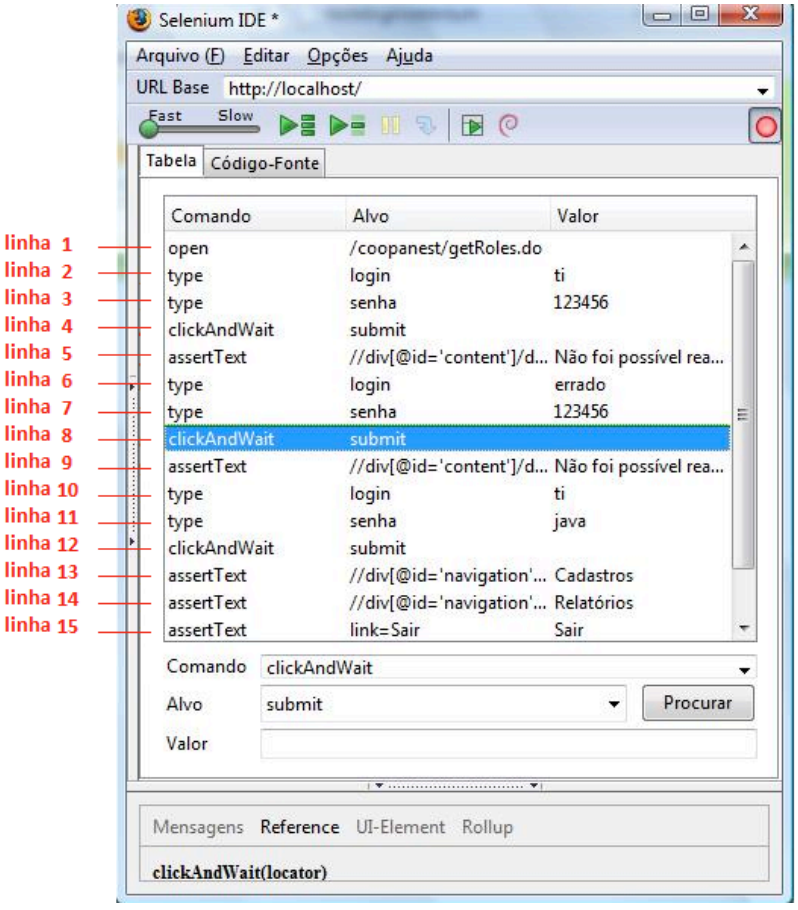


Figura 1. Exemplo de teste funcional no Selenium.

Testes de estresse geralmente são realizados de forma conjunta com testes de desempenho, não sendo raro haver confusão entre os mesmos. Enquanto o teste de desempenho tem como objetivo testar se determinado sistema é capaz de lidar com a carga esperada e definida nos requisitos, o teste de estresse consiste em submeter o sistema a situações anormais de uso [1], como grandes quantidades de carga, redução dos recursos computacionais disponíveis e entradas não realistas de dados [7]. Embora a maioria dos sistemas não responda às requisições quando submetidos a situações de estresse, a partir dessa sobrecarga pode-se descobrir um comportamento indesejável, como por exemplo, o vazamento de informações confidenciais de um banco de dados em mensagens de erro [8].

A partir dos testes de estresse torna-se possível observar o comportamento do sistema durante situações críticas, identificando falhas não toleráveis e potencialmente difíceis de serem encontradas em situações normais de funcionamento. Um exemplo disso é o caso do Sistema de Emergência de Londres, citado anteriormente. O sistema funcionou corretamente durante algum tempo, porém, ao ser exposto a uma carga maior, ele respondeu em um tempo acima do tolerável, causando problemas no roteamento das ambulâncias e, a partir disso, demora no atendimento, o que culminou com a morte de pessoas [3].

A Figura 2 exibe um trecho de código gerado pelo JMeter[4] para um teste de desempenho para o mesmo exemplo utilizado para os testes funcionais da Figura 1: o *login* em uma aplicação. Ele foi criado para verificar se a aplicação suporta 50 usuários simultâneos (propriedade *ThreadGroup.num_threads*), com tempo de resposta máximo de 3 segundos (propriedade *DurationAssertion.duration*). O restante do código ilustrado refere-se à formação das URLs que serão utilizadas no teste, com seus respectivos dados de entrada. No teste de desempenho normalmente é verificado o "caminho feliz" da aplicação, ignorando-se as diferentes condições alternativas existentes. Além disso, normalmente não existe uma preocupação excessiva com as respostas obtidas e sim se houve resposta dentro do contexto do teste. Para se criar um teste de estresse a partir do teste de desempenho exibido, basta elevar um parâmetro do teste, como por exemplo, o número de usuários simultâneos, criando um contexto acima do estabelecido como aceitável para a aplicação[1].

Conforme foi brevemente discutido nesta seção, os testes funcionais e testes de desempenho e estresse são bem diferentes, possuindo diferentes ferramentas, linguagens e aspectos a analisar. No entanto, pode-se observar que os testes funcionais podem ser uma fonte de dados para a geração dos testes de desempenho e estresse. Essa foi a idéia explorada neste trabalho, conforme discutido nas próximas seções.

[4] http://jakarta.apache.org/jmeter/

```
<testelement>
  <property name="TestElement.name">Teste de Login</property>
</testelement>

<testelement>
  <property name="ThreadGroup.num_threads">50</property>
  <property name="ThreadGroup.ramp_time">1</property>
</testelement>

<testelement>
  <property name="HTTPSampler.path">${contexto}autentication.do </property>
  <property name="HTTPSampler.method">GET</property>
  <property name="TestElement.name">Login</property>
</testelement>

<testelement name="HTTPsampler.Arguments">
  <collection name="Arguments.arguments">
    <testelement name="">
      <property name="Argument.name">login</property>
      <property name="Argument.value">ti</property>
    </testelement>
    <testelement name="">
      <property name="Argument.name">senha</property>
      <property name="Argument.value">java</property>
    </testelement>
  </collection>
</testelement>

<testelement>
  <property name="DurationAssertion.duration">3000</property>
  <property name="TestElement.name">Tempo maximo de resposta</property>
</testelement>
```

Figura 2. Exemplo de teste de desempenho no JMeter.

3 Geração de Testes de Desempenho e Estresse a partir de Testes Funcionais

Neste trabalho foi desenvolvido um método e uma ferramenta para a geração de testes de desempenho e estresse a partir de testes funcionais. Esta seção descreve o "método" ("método" na acepção de Houaiss e Villar [9]), que corresponde ao conjunto de passos para efetiva geração dos testes de requisitos não funcionais. A maioria desses passos é feita diretamente a partir do uso da ferramenta, porém, não todos. O conjunto de passos é descrito a seguir.

1. Criação dos testes funcionais. Neste primeiro passo devem ser criados os testes funcionais para o sistema sob teste utilizando alguma ferramenta de teste funcional. Esses testes serão os artefatos básicos para a geração do teste de desempenho e estresse. Conforme comentado na Seção 2, a Figura 1 exibe um teste funcional criado com o Selenium IDE. No exemplo ilustrado, o teste é composto do acesso a uma página (*open*), da entrada de valores nos campos

exibidos nessa página (*type*), da execução de comandos (*clickAndWait*) e da verificação dos resultados obtidos (*AssertTextPresent*).

2. Extração dos procedimentos e casos de teste relacionados. Analisando o teste funcional é possível obter as ações relacionadas (procedimentos de teste), assim como os dados de entrada, dados de saída e demais condições associadas ao teste (caso de teste). Essa extração pode ser feita automaticamente a partir do uso da ferramenta construída.

3. Informação sobre os requisitos específicos da aplicação. Uma aplicação pode ter requisitos que dificultem a geração de testes de desempenho e estresse. Em sistemas que permitem a criação de usuários, por exemplo, é muito provável que não seja permitido a criação de usuários com identificadores repetidos. Logo, testes de desempenho e estresse para esse sistema necessitam de vários usuários simultaneamente, cada um com identificador único. Além da unicidade, pode haver restrições quanto ao formato e tamanho dos dados de entrada, por exemplo. Como não é possível obter tais informações diretamente dos testes funcionais, esses requisitos devem ser fornecidos manualmente e incorporados de alguma forma aos procedimentos e casos de testes extraídos, obtendo como resultado um único arquivo com todas essas informações. No caso da ferramenta construída, isso é feito através de uma tela específica para informação dessas restrições.

4. Informação dos requisitos de desempenho e estresse. Neste passo devem ser informados, pelo testador, os requisitos de desempenho e estresse. São exemplos desses requisitos: número máximo de usuários concorrentes e o tempo máximo de resposta para as operações. Além disso, a forma como o teste vai ser executado deve ser especificada. Deve-se definir, por exemplo, o número de máquinas a serem usadas no teste e o tempo entre cada requisição.

5. Geração do teste de desempenho e estresse. O último passo do método consiste na utilização das informações extraídas do teste funcional e das informações fornecidas pelo testador para a geração do teste de desempenho ou estresse, o qual poderá ser executado em uma ferramenta de teste específica. No caso da ferramenta construída, basta indicar o formato a ser utilizado e solicitar a geração do teste de desempenho ou estresse.

4 FERRARE: a ferramenta protótipo desenvolvida

Para concretizar a idéia proposta, foi desenvolvida uma ferramenta protótipo que permite a geração de testes de desempenho e estresse a partir de testes funcionais. A ferramenta em questão foi denominada de FERRARE - **FERR**amenta de **A**utomação dos testes de **R**equisitos de desempenho e **E**stresse. Ela foi concebida para trabalhar com qualquer ferramenta de teste funcional e qualquer ferramenta de teste de desempenho e

estresse, desde que sejam implementadas as interfaces apropriadas, conforme será discutido mais adiante. Atualmente, a FERRARE trabalha com as ferramentas de testes funcionais Selenium IDE e Canoo Web Test[5], além das ferramentas de teste de desempenho e estresse Apache JMeter e WebLoad[6]. Isso significa que ela gera testes de desempenho e estresse para serem executados no JMeter ou WebLoad, a partir de testes funcionais feitos com o Selenium IDE ou Canoo WebTest.

Parte da estrutura interna da FERRARE é exibida na Figura 3. A FERRARE é dividida em dois módulos: Extrator e Gerador. O módulo Extrator é responsável pela extração das informações contidas no teste funcional. Isso inclui a criação de um procedimento de teste e identificação dos dados de entrada, saídas esperadas e demais condições do teste, que irão compor o caso de teste. O Gerador é responsável pela geração do teste de desempenho seguindo um formato apropriado.

Conforme mencionado, a FERRARE foi desenvolvida para ser utilizada com qualquer ferramenta de teste funcional, de desempenho ou de estresse. Para isso, foram criados interfaces e classes abstratas que devem ser implementadas para adequação à uma ferramenta específica. Para a utilização do Selenium como fonte dos testes funcionais, foi necessário desenvolver o *SeleniumExtractor*, que estende a classe *BaseExtractor*, que possui alguns métodos abstratos a serem definidos pelos Extratores. O principal método do *BaseExtractor* é o *getNextTest*, que retorna um teste funcional. Normalmente, a implementação desse método é feita a partir de um *parser* que analisa o teste e gera sua representação interna utilizando o formato especificado na FERRARE, que inclui a determinação das entradas, saídas e ações associadas, na forma dos procedimentos de teste, conforme apresentado na Figura 3.

O *SeleniumExtractor*, por exemplo, estende a classe *BaseExtractor* e implementa o método *getNextTest*. A implementação do *SeleniumExtractor* leva em consideração os comandos existentes no Selenium. Assim, o comando "*assert"*, que é utilizado para verificar a presença de um texto, está associado com as saídas esperadas do caso de teste; o comando "*type"*, utilizado para inserir um dado em um campo da página, está associado às entradas do caso de teste; o comando "*click"*, utilizado para se clicar em um elo ou botão; e o comando "*open*", utilizado para acessar um endereço, são utilizados para executar ações relacionadas ao teste. Logicamente, em outras ferramentas existirão outros comandos, com finalidades iguais ou semelhantes as dos comandos do Selenium. Cabe ao Extrator abstrair essas diferenças e retornar um teste no formato exigido pela FERRARE, que ignora aspectos irrelevantes para a geração do teste de desempenho e estresse.

[5] http://webtest.canoo.com/
[6] http://www.webload.org/

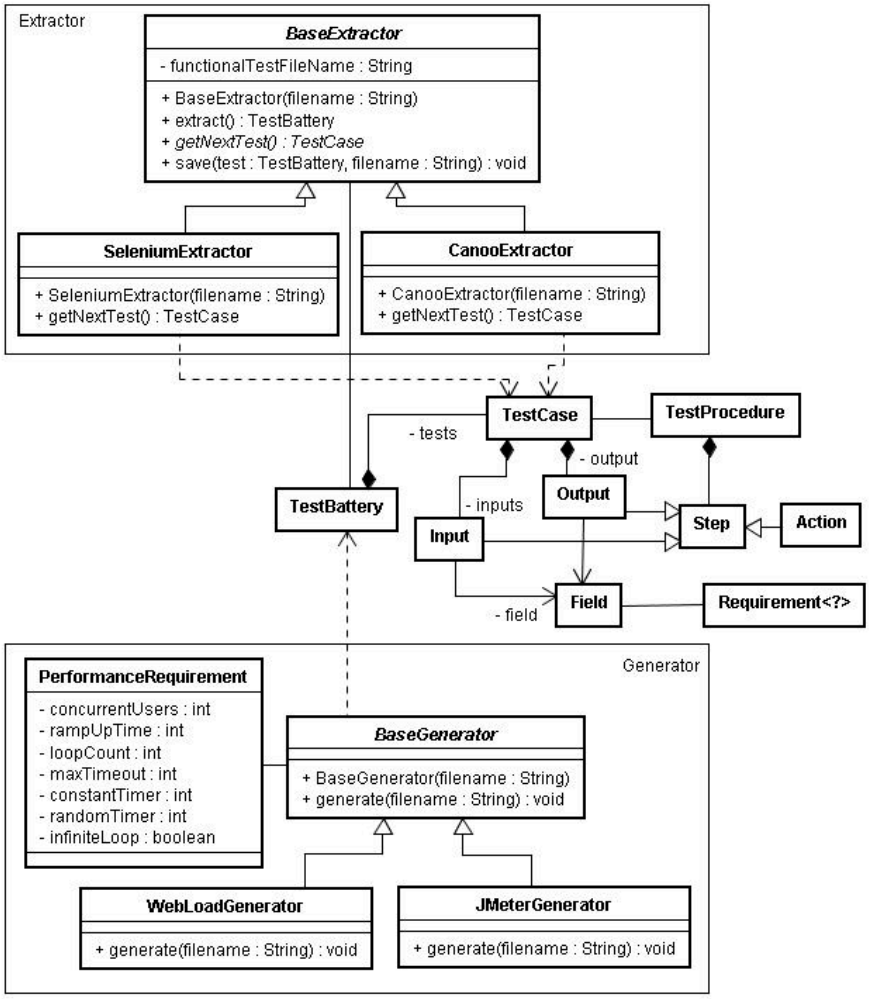


Figura 3. Principais classes existentes na FERRARE.

O outro módulo da FERRARE, o Gerador, tem por objetivo gerar os testes de desempenho e estresse a partir da representação interna do teste funcional gerada pelo Extrator. Para isso, a ferramenta cria uma representação interna do teste de desempenho ou estresse, permitindo ainda uma especificação dos requisitos associados ao teste, como por exemplo, o número de usuários concorrentes e tempo máximo de resposta de requisições, conforme apresentado na Figura 3. Assim como o Extrator, essa parte da ferramenta é maleável e pode ser facilmente estendida para o uso com outras ferramentas de testes além daquelas já suportadas (JMeter e WebLoad). Existe uma classe abstrata, denominada

BaseGenerator, que tem acesso às informações extraídas pelo Extrator e que deve ser utilizada para geração dos testes de desempenho e estresse. Cada Gerador específico deve ser criado levando em consideração a linguagem da ferramenta utilizada para geração dos testes. O Gerador para uma ferramenta deve implementar o método *generate*, que deve gerar os testes de desempenho e estresse com base nos testes abstratos (extraídos do teste funcional e independentes de tecnologia) e nos requisitos de desempenho e estresse. Assim, caso alguém deseje criar um novo Gerador, basta criar uma classe que herde de *BaseGenerator* e que siga as instruções relatadas aqui.

A FERRARE ainda possui limitações atualmente. Uma delas diz respeito a não identificação de todas as URLs acessadas no teste funcional, pois, em geral, ferramentas de testes funcionais gravam apenas as ações que estão sendo executadas, simulando um usuário da aplicação. Como essas URLs não estão disponíveis no teste funcional, a geração dos testes de desempenho e estresse é dificultada, pois estes necessitam destas URLs para montar as requisições que serão feitas à aplicação. A solução encontrada para este problema foi deixar para a ferramenta de teste de desempenho e estresse a tarefa de encontrar as URLs acessadas. Assim, quando o Gerador cria o teste de desempenho e estresse, ele indica quais URLs precisam ser descobertas e utiliza os recursos da ferramenta de destino para isso. No caso do JMeter, por exemplo, utilizam-se das suas expressões regulares para analisar a página e descobrir qual a URL adequada para a próxima requisição que será feita.

Outra limitação diz respeito à não geração dos dados necessários à execução dos testes de desempenho e estresse criados pela FERRARE. Dessa forma, embora o teste seja gerado pela ferramenta construída, um esforço significativo ainda é necessário para criar os dados requeridos, bem como para adequar o teste gerado à utilização desses dados. Para melhor entendimento dessa limitação, suponha-se, por exemplo, a geração de testes de desempenho a partir de um teste funcional que realiza um empréstimo de livro em um sistema de biblioteca on-line. Um teste para verificar 100 empréstimos de livros ao mesmo tempo não poderia ser gerado automaticamente pela ferramenta, uma vez que envolve uma combinação de condições para um único empréstimo, tais como a existência de um livro disponível (que não seja cativo e possa ser emprestado) e de um usuário com menos de cinco livros em empréstimo (supondo que cinco seja o limite máximo para a categoria do usuário em questão). É importante frisar que esse exemplo é de difícil implementação em qualquer ferramenta, uma vez que requer uma grande quantidade de dados iniciais. Para restrições mais simples a FERRARE trabalha sem qualquer problema

Para resolver esse problema, encontra-se em desenvolvimento a GÊNESIS, um novo módulo para a FERRARE que gera dados a partir de um modelo pré-existente. Com isso, será utilizado como modelo a especificação de um caso de teste válido. A partir disso é realizado uma cópia dos dados necessários ao teste, levando-se em consideração as restrições associadas. Mais detalhes sobre o GENESIS e sobre a geração de dados a partir de testes funcionais pode ser encontrada no artigo que descreve o protótipo já implementando [10]. Outra alternativa seria especificar as restrições na ferramenta, a partir da modelagem de restrições, mas isso pode ser bem mais complexo e se distancia da idéia original do trabalho, que tem como objetivo não prescrever a criação de nenhum artefato adicional.

5 Estudo Experimental

Foi realizado um estudo experimental com o intuito de comparar o tempo necessário para a criação de testes de desempenho e estresse com e sem o uso do método e da ferramenta desenvolvida. O propósito do estudo foi avaliar, com respeito a esforço, do ponto de vista do pesquisador, no contexto de alunos de graduação em Ciência da Computação, a aplicabilidade do método e da FERRARE na criação de testes de desempenho e estresse para um sistema *Web*.

No experimento foram utilizados o *SeleniumExtractor* e o *JMeterGenerator*. Por conta disso, a principal questão relacionada ao experimento foi: *a utilização do método e da FERRARE gera uma redução no esforço necessário para a criação dos testes de desempenho e estresse para serem executados com o JMeter, quando comparado à criação dos mesmos testes utilizando somente o JMeter, sem o apoio da FERRARE*? A hipótese nula, relacionada a essa questão é: não existe diferença no esforço, medido em termos de minutos, para criar os testes com e sem o apoio do método e da ferramenta, ou seja, H0: EsforçoTeste(FERRARE) = EsforcoTeste(JMeter). A hipótese alternativa é que o esforço empregado no teste, com apoio do método é menor que o esforço empregado sem o uso do método e da ferramenta, ou seja, EsforçoTeste(FERRARE) < EsforcoTeste(JMeter). No entanto, conforme será detalhado a seguir, o desenho experimental utilizado permitiu que fosse avaliada uma série de outras questões relevantes ao estudo.

Foi utilizada a ferramenta Selenium pelo fato dela ser uma das ferramentas mais comentadas nas revistas técnicas existentes no Brasil. Além disso, ela possui boa usabilidade, é livre, gratuita e se mostrou bastante adequada em todas as avaliações que foram executadas pelo grupo de pesquisa. Na avaliação experimental foram entregues testes funcionais feitos utilizando Selenium para que os participantes pudessem gerar testes não funcionais com a FERRARE.

Foram utilizados como participantes alunos das disciplinas de Engenharia de *Software* do curso de Bacharelado em Ciência da Computação da Universidade Federal do Piauí (UFPI). Esses alunos cursaram as disciplinas de Engenharia de *Software* ofertadas no ano de 2008. Participaram 23 voluntários, dos quais 20 concluíram todas as etapas previstas. Os participantes não tinham experiência alguma no uso de ferramentas de automação de testes funcionais e nem no uso de ferramentas para o teste de desempenho e estresse. Por conta disso não foi feito nenhum tipo de agrupamento baseado no perfil dos mesmos. Antes do início do estudo foi realizada uma breve apresentação com relação às atividades que seriam realizadas. Também foi garantido que seria preservado o anonimato dos estudantes explicando ainda como os dados coletados seriam utilizados. Todos os voluntários concordaram com o experimento e permitiram sua utilização para fins acadêmicos.

O desenho experimental utilizado levou em consideração as possíveis ameaças à validade do mesmo. A Figura 4 resume o desenho experimental utilizado. Conforme pode ser visualizado na figura, todos os participantes tiveram contato com as duas ferramentas, porém em momentos diferentes e realizando exatamente as mesmas tarefas. Isso permitiu

que um grupo atuasse como controle do outro. Assim, os participantes do Grupo 1 (G1) criaram testes utilizando o JMeter e só depois criaram os mesmos testes utilizando a FERRARE. O Grupo 2 (G2) iniciou criando os testes com a FERRARE e depois criaram testes com o JMeter. A atribuição aos grupos foi aleatória. A seleção dos participantes foi baseada na conveniência, por isso o estudo é considerado um quase-experimento [11]. Vale ressaltar que em todos os casos os testes eram executados no JMeter, ou seja, a diferença se concentrava em como os testes eram criados: se utilizando a FERRARE ou o próprio JMeter.

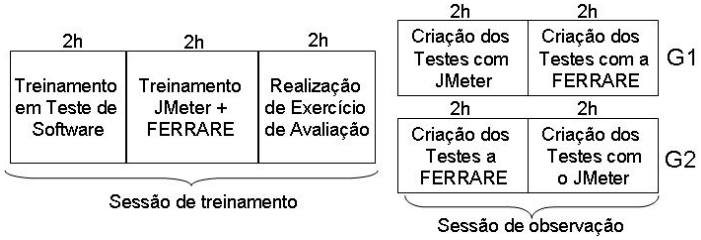


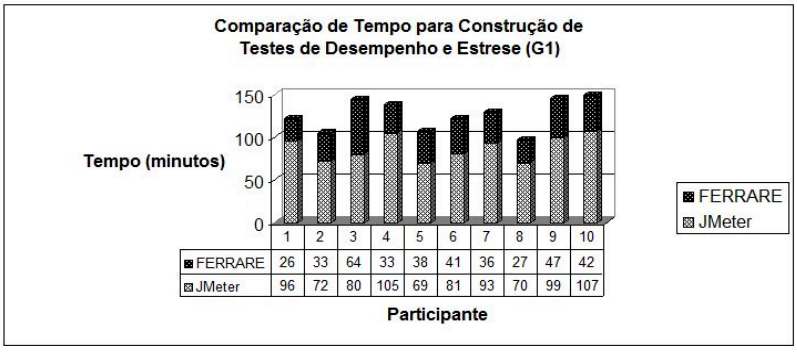
Figura 4. Desenho experimental utilizado.

Em termos de implementação, o estudo foi dividido em duas sessões, uma envolvendo apenas o treinamento das ferramentas que seriam utilizadas com apoio dos autores deste artigo e outra sessão dedicada para observação, sem interferência dos autores. Durante o treinamento foi apresentado aos participantes uma visão geral sobre testes de *software* e em seguida foram detalhadas as ferramentas JMeter e FERRARE. A partir disso foram realizados alguns exercícios para a verificação do aprendizado, com o apoio dos autores do artigo, dirimindo dúvidas e auxiliando os voluntários naquilo que fosse necessário. A idéia era treiná-los de forma adequada para que durante a observação não fosse necessária nenhuma intervenção externa, uma vez que isso poderia comprometer os dados coletados.

Durante a sessão de observação, os participantes do estudo tiveram que criar testes de desempenho para parte de um sistema gerenciador do sítio da UFPI. A função desse sistema é tornar simples a criação de páginas para o sítio, seguindo formatos pré-estabelecidos. Os usuários do sistema podem criar menus, notícias, destaques, usuários para administração de páginas e elos para outras páginas, além de criar subpáginas, que funcionam tal qual a página principal e com todas as funcionalidades anteriormente descritas. Existem oito casos de uso nesse sistema, dos quais dois foram utilizados durante a sessão de treinamento: Gestão de Usuários e Gestão de Eventos; e outros dois foram utilizados na sessão de observação: Gestão de Notícias e Gestão de Menus. Todos os casos de uso são implementações de CRUD (*create, read, update, delete*).

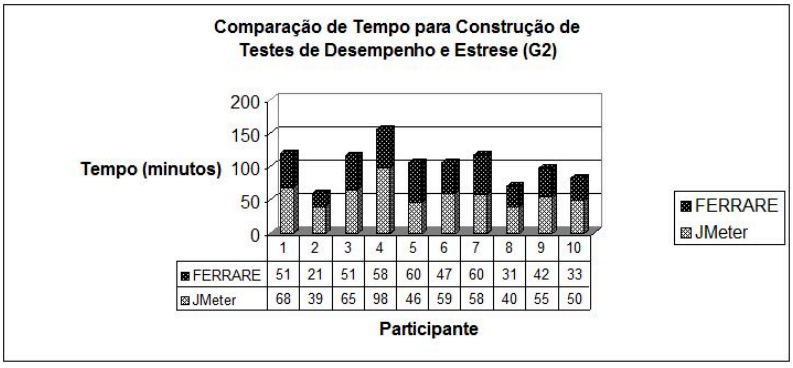
As Figuras 5 e 6 apresentam os dados colhidos no estudo. A Figura 5 apresenta os dados relativos à execução do estudo iniciando pelo uso do JMeter (G1), enquanto que a Figura 6 apresenta os dados do grupo que iniciou com a FERRARE (G2). Analisando esses dados pode-se notar que os participantes utilizando o método proposto neste trabalho dedicaram um esforço menor para a criação de testes de desempenho e estresse que os

participantes que criaram tais testes sem o apoio do método e da FERRARE, em ambos os casos. A explicação para isso está no fato da FERRARE automatizar a geração do teste de desempenho e estresse, utilizando os testes funcionais como base para essa criação. Com isso, o usuário da ferramenta precisa apenas especificar alguns parâmetros para o teste de desempenho e estresse, reduzindo o tempo gasto para a criação dos mesmos. É importante ressaltar que todos os testes desenvolvidos foram executados pelos participantes, para homologar o trabalho realizado, e verificados pelos autores do artigo, para garantir que as especificações de teste foram desenvolvidas em sua totalidade.



	1	2	3	4	5	6	7	8	9	10
FERRARE	26	33	64	33	38	41	36	27	47	42
JMeter	96	72	80	105	69	81	93	70	99	107

Figura 5. Comparação de tempos para a construção dos testes do Grupo 1 (JMeter x FERRARE).



	1	2	3	4	5	6	7	8	9	10
FERRARE	51	21	51	58	60	47	60	31	42	33
JMeter	68	39	65	98	46	59	58	40	55	50

Figura 6. Comparação de tempos para a construção dos testes do Grupo 2 (FERRARE x JMeter).

A Tabela 1 apresenta os resultados dos testes de hipóteses para cada uma das questões associadas ao estudo experimental, com seus respectivos níveis de significância, graus de liberdade e escore t de referência. Conforme foi descrito anteriormente, o desenho experimental utilizado permitiu que fossem trabalhadas algumas questões interessantes

relacionadas ao estudo, conforme detalhamento a seguir. Em cada uma das questões foi utilizado o *teste t de Student* para realizar o teste de hipóteses.

Tabela 1. Principais questões associadas ao estudo experimental.

Nr.	Comparação	Teste t	α	f	t α/2, f
1	Teste t para comparar todos os tempos para criação dos testes: Jmeter (G1 e G2) x FERRARE (G1 e G2).	5,46	5%	18	2,101
2	Teste t pareado para comparar os tempos para criação dos testes no Grupo 1 (JMeter x FERRARE).	8,51	5%	9	2,262
3	Teste t pareado para comparar os tempos para criação dos testes no Grupo 2 (FERRARE x JMeter).	2,81	5%	9	2,262
4	Teste t para comparar os tempos para criação dos testes com JMeter em G1 e G2.	4,13	5%	18	2,101
5	Teste t para comparar os tempos para criação dos testes com a FERRARE em G1 e G2.	1,22	5%	18	2,101
6	Teste t para comparar os tempos para criação dos testes com JMeter em G1 e FERRARE em G2 (primeira tarefa nos dois grupos).	6,68	5%	18	2,101
7	Teste t para comparar os tempos para criação dos testes com JMeter em G2 e FERRARE em G1 (segunda tarefa nos dois grupos).	2,96	5%	18	2,101

A principal questão do experimento está expressa na primeira linha da tabela, e que pode ser resumida da seguinte forma: os tempos para criar os mesmos testes, utilizando o JMeter e a FERRARE, independentemente de ordem (considerando todos os tempos nos dois grupos) são estatisticamente iguais? Foi utilizado o Teste t para essa comparação. O resultado, presente na Tabela 1, indica que os tempos são significativamente diferentes. Esse resultado era esperado, uma vez que a FERRARE automatiza boa parte da geração do teste, exigindo apenas a especificação de alguns parâmetros para essa geração. É importante ressaltar que foram realizados experimentos utilizando dois grupos, com ordens de execução dos passos do experimento diferentes. Essa comparação envolveu todos, independente da ordem utilizada e apresentou como resultado a indicação que o uso da FERRARE reduz o esforço necessário para criação dos testes.

Outra questão verificada a partir do experimento foi se o resultado também ocorreu quando se compara os tempos para criar os mesmos testes em uma ordem específica. Essa análise é interessante para se descobrir se a ordem de execução das tarefas pode criar algum viés ao experimento. Os resultados indicam que a FERRARE gera benefícios, independentemente da ordem utilizada. Os dados relacionados a essa questão, comparando tanto os tempos na ordem FERRARE (G1) x JMeter (G1), assim como JMeter (G2) x FERRARE (G2), FERRARE (G2) x JMeter (G1) e JMeter (G2) x FERRARE (G1) são apresentados, respectivamente nas linhas 2, 3, 6 e 7 da Tabela 1.

Também foi verificado se a utilização de uma ferramenta influencia no uso da outra. O uso da FERRARE, seja antes ou depois de ter sido utilizado o JMeter, não apresentou diferenças significativas (linha 5). No entanto, foi possível notar que o uso do JMeter, após o

uso da FERRARE gerou tempos significativamente menores (linha 4). De acordo com esse resultado, usar o JMeter depois de ter usado a FERRARE gera ganhos no tempo para realização da tarefa, indicando assim que os desenvolvedores que utilizam JMeter se tornam mais produtivos. Por conta disso, os participantes do experimento foram entrevistados para obtenção de maiores dados sobre esse fenômeno. Foi unânime a explicação para o fato: como a FERRARE gera os testes em JMeter e eles tiveram que colocá-lo em execução para homologar seus trabalhos (podendo assim visualizar sua estrutura), o aprendizado obtido com isso influenciou de forma positiva a realização da criação manual desses testes. A visualização do teste criado pela FERRARE torna a tarefa de geração dos testes em JMeter bem mais fácil. Mesmo assim, é importante ressaltar que o uso da FERRARE ainda gerou ganhos substanciais em termos de tempo, mesmo sendo comparado com esse viés favorecendo a criação de testes com o JMeter.

Em resumo, todas as comparações que foram realizadas mostraram que o uso da FERRARE gera ganhos significativos em termo de esforço. No entanto, é importante analisar o quão válidos são os resultados obtidos. Idealmente, os resultados deveriam ser válidos não somente na população em que eles foram recebidos, mas também para uma população mais ampla. Os resultados possuem a validade adequada se são válidos para a população a qual tendem a ser generalizados. No caso de experimentos em Engenharia de Software, normalmente deve ser avaliado a Validade Interna e a Validade Externa, que são as mais importantes nesse tipo de estudo [12].

A validade interna define se o relacionamento observado entre o tratamento e o resultado é causal e não influência de outros fatores não controlados. De acordo com os resultados obtidos, o uso da FERRARE gera uma redução no esforço necessário para desenvolvimento de testes. Como os participantes tiveram que automatizar testes para a mesma parte do sistema, utilizando tanto a FERRARE quanto o JMeter, não foi notado o problema da *instrumentação*, uma vez que não há diferença no problema utilizado no estudo.

Outro problema que poderia influenciar os resultados é a ameaça conhecida como *testagem*. Ela se refere ao fato da pessoa ficar mais produtiva entre as tarefas, por conhecer mais o sistema e o próprio problema a ser tratado. Conforme foi enfatizado anteriormente, a *testagem* influencia apenas o desenvolvimento de testes com o JMeter, uma vez que a FERRARE gera testes para o JMeter, permitindo assim que os participantes aprendam mais sobre o JMeter, reduzindo o tempo total para criação de testes nessa ferramenta, após o uso da FERRARE. No entanto, mesmo com esse viés favorecendo a produtividade dos voluntários utilizando o JMeter, foi possível constatar que os participantes utilizando a FERRARE foram mais produtivos, o que comprova que a ferramenta realmente gera ganhos em termos de produtividade.

Nada relacionado ao estudo indica que houve alguma ameaça relacionada à *história*, ou seja, não foi identificado nenhum efeito externo que pudesse influenciar nos resultados diferentemente do que foi mencionado acima. Com relação à *maturação*, acredita-se que os participantes melhoraram com o tempo, uma vez que são alunos e estão submetidos a

diversos conteúdos novos, mas nada que pudesse influenciar diretamente o estudo desenvolvido.

O desenho experimental escolhido, em que todos os participantes utilizaram as duas ferramentas permitindo que um grupo atuasse como controle do outro, validam a conclusão obtida, ao mesmo tempo em que minimizam qualquer ameaça relacionada ao *comportamento competitivo* e ao *comportamento compensatório* [12].

A validade externa define as condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial. Os participantes do estudo, alunos do 6º e 7º período do curso, encontram-se em fase final de formação, possuindo um perfil próximo a profissionais com pouca experiência. Além disso, os participantes nunca trabalharam com ferramentas de automação de testes, mas 55% deles afirmaram ter experiência de trabalho como programadores. Os treinamentos ministrados, juntamente com o exercício de fixação contribuíram para uma boa formação dos participantes nos objetos (FERRARE e JMeter) do estudo. O exemplo utilizado, dois casos de uso de um sistema para *Web*, embora seja pequeno, possui as características comumente existentes nos sistemas de informação para *Web*. Assim, acredita-se que as conclusões obtidas no estudo possam ser estendidas para outros sistemas, com tamanho usual e utilizados por profissionais com pouca experiência, sem prejuízos nos resultados observados.

6 Trabalhos Relacionados

Existem poucos trabalhos relacionados a testes de desempenho e estresse, em especial, tratando dos aspectos de automação. Por conta disso, um interessante trabalho nessa área foi desenvolvido por Weyuker e Vokolos [6]. Eles descrevem questões essenciais relacionadas a testes de desempenho, ressaltando a importância desses testes. Os autores exploram a idéia de elaborar casos de testes específicos para testar o desempenho do sistema ao invés de corretude funcional e discorrem sobre a criação de cargas de trabalho representativas para os sistemas em teste, além de diversos outros aspectos essenciais.

Boa parte dos trabalhos que envolvem automação dos testes de desempenho e estresse faz isso a partir de modelos descrevendo o *software*. Um artefato muito utilizado como base para essa automação são as máquinas de estados finitos. O projeto AGEDIS [13] inclui uma ferramenta que pode automatizar testes de desempenho e estresse a partir da descrição de um modelo comportamental do *software*, utilizando máquinas de estado. Shams, Krishnamurty e Far [14] descrevem uma abordagem baseada em modelos para o teste de desempenho em aplicações *Web*, utilizando máquinas de estado para modelar as dependências entre requisições e as dependências de dados de um sistema. O grande problema das abordagens utilizando máquinas de estado, descrito no relatório final do projeto AGEDIS, é que tal forma de modelagem não parece ser apropriada para representar interações entre diversos elementos, gerando um esforço muito grande e baixa inteligibilidade.

Outras abordagens para automação utilizam modelos baseados em padrões de mercado, como a UML. Garousi, Briand e Labiche [7] desenvolveram uma metodologia de teste de estresse em sistemas distribuídos baseada em modelos UML estendidos com informações sobre tempo e análise de controle de fluxo em diagramas de seqüência. O problema nessa abordagem é propor a criação de mais artefatos durante o desenvolvimento, gerando ainda mais trabalho e "burocracia", uma vez que a tarefa de mantê-los consistentes e atualizados não é trivial.

De forma geral, os trabalhos mencionados descrevem métodos e/ou ferramentas que podem gerar testes de desempenho e estresse a partir de uma descrição do sistema. Neste trabalho é apresentada uma idéia inovadora: utilizar os próprios testes funcionais para gerar os testes de desempenho e estresse. E mais do que a geração de outros testes, descobriu-se que é possível se utilizar das informações dos testes funcionais para gerar dados e até mesmo documentos, conforme descrito em detalhes em [10] e [15], respectivamente.

7 Considerações Finais

A atividade de teste é parte importante do processo utilizado para garantir que os requisitos identificados pelos usuários sejam atendidos pelo produto desenvolvido. Os testes, entretanto, requerem tempo, planejamento, infra-estrutura e conhecimento especializado, sendo, portanto, uma atividade inerentemente onerosa e que exige profissionais qualificados. Embora os testes funcionais estejam relativamente bem difundidos nas organizações, isso não acontece com os testes de desempenho e estresse, que são bem menos difundidos. A principal explicação relacionada a esse fato é a ausência de profissionais qualificados e o alto custo para realização dessa tarefa.

Neste trabalho foi apresentado um método e uma ferramenta para auxiliar a difusão do teste de desempenho e estresse. É importante ressaltar que os testes de desempenho e estresse estão crescendo em importância, uma vez que os sistemas para *Web* cada vez mais ganham espaço em relação aos sistemas *desktop*, no entanto, existem poucos profissionais qualificados para a criação desses testes. Como os testes funcionais encontram-se razoavelmente bem difundidos nas organizações e possuem praticamente todas as informações necessárias para a geração automática de testes de desempenho e estresse, eles foram utilizados para automação desses testes. O principal objetivo desta idéia é tornar mais simples a criação de testes de desempenho e estresse, não exigindo tanto conhecimento na criação desses testes e, ao mesmo tempo, reduzindo o esforço relacionado. Isso foi confirmado no estudo experimental que foi realizado: os participantes que utilizaram o método e a ferramenta foram mais efetivos, em termo de esforço, para gerar o mesmo conjunto de teste. O estudo foi idealizado de forma a contrapor diversas ameaças à sua validade. Os resultados obtidos foram positivos em relação ao uso da FERRARE.

Além disso, os testes de desempenho e estresse gerados com base em testes funcionais tendem a cobrir toda a aplicação sendo testada, uma vez que essa geração simplifica muito a tarefa do testador. Dados os custos associados, é muito comum em

organizações que realizam o teste de desempenho e estresse focar em apenas alguns casos de uso e ignorar outros. O apoio da ferramenta descrita neste trabalho reduz o custo para o teste de desempenho e estresse, tornando plenamente viável a cobertura de todo o sistema. Embora a qualidade dos testes gerados não tenha sido uma variável no estudo, pois foi exigido de todos os participantes um nível de qualidade padrão, uma vez que todos deveriam implementar uma especificação de teste previamente criada, é fácil notar que um testador utilizando a FERRARE certamente gerará testes com maior cobertura, uma vez que ele pode fazer isso em bem menos tempo. É importante ressaltar que a FERRARE auxilia a implementação de testes e não seu projeto. O projeto é que define a qualidade dos testes. Por isso, esse fator não foi diretamente abordado no estudo.

Outro fator interessante é que a ferramenta é facilmente extensível, podendo ser adaptável para qualquer conjunto de ferramentas. Basta para isso implementar as interfaces e classes abstratas criadas para facilitar essa extensão, conforme comentado na Seção 4. É igualmente importante ressaltar que a idéia explorada neste trabalho não foi encontrada em nenhum outro trabalho relacionado. Conforme foi discutido na Seção 6, existem muitas idéias associadas à facilitação da geração de testes de desempenho e estresse, porém sem explorar a reutilização de testes funcionais. Essa abordagem tem se mostrado muito promissora e será continuada, uma vez que os resultados apresentam dados interessantes. Um dos trabalhos em andamento é o estudo da expressividade da abordagem e a resolução da limitação da ferramenta em relação à existência de restrições complexas.

Referências

[1] Myers (2004), G. *The Art of Software Testing*. John Wiley & Sons, 2ª edição.

[2] Hutcheson, M. L. (2003) *Software Testing Fundamentals: Methods and Metrics*. John Wiley & Sons.

[3] Finkelstein, A. e Dowell, J. (1996). A comedy of errors: the London ambulance service case study. In IWSSD'96: *Proceedings of the 8th International Workshop on Software Specification and Design*, page 2, Washington, DC, USA. IEEE Computer Society.

[4] MCT (2007). Programa Brasileiro da Qualidade e Produtividade de Software. Ministério da Ciência e Tecnologia, 4ª edição.

[5] Denaro, G.; Polini, A.; e Emmerich, W. (2004). Early performance testing of distributed software applications. *SIGSOFT Software Engineering Notes*, 29(1):94-103.

[6] Weyuker, E. e Vokolos, F. (2000). Experience with performance testing of software systems: Issues, an approach, and case study. *IEEE Transactions on Software Engineering*, 26(12):1147-1156.

[7] Garousi, V., Briand, L., e Labiche, Y. (2006). Traffic-aware stress testing of distributed systems based on UML models. *In Proceedings of the 28th International Conference on Software Engineering (ICSE)*, pages 391-400, Shangai, China.

[8] Sommerville, I. (2003). *Engenharia de Software*. Addison Wesley, 6ª edição.

[9] Houaiss, A. e Villar, M. (2001). *Dicionário Houaiss de língua portuguesa*.

[10] Sousa Fé, I.; Santos, A. R.; Santos, I. S.; Santos Neto, P.; Britto, R. S. (2010). Geração de Dados para Testes de Desempenho e Estresse a Partir de Testes Funcionais. In: IX *Simpósio Brasileiro de Qualidade de Software*, Belém, PA. Anais do IX Simpósio Brasileiro de Qualidade de Software, 2010. p. 89-102.

[11] Wainer, J. (2007). Pesquisa quantitativa e qualitativa em ciência da computação. In Kowaltowski, T. and Breitman, K., editors, *Jornada de Atualização em Informática (JAI)*, pág. 221-262. Sociedade Brasileira de Computação (SBC), Editora PUC Rio.

[12] Wohlin, C.; Runeson, P.; Host, M.; Ohlsson, M.; Regnell, B.; e Wesslen, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publihers.

[13] Hartman, A. e Nagin, K. (2004). The AGEDIS tools for model based testing. *In Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 2004)*, Boston, Massachusetts, USA.

[14] Shams, M.; Krishnamurthy, D.; e Far, B. (2006). A model-based approach for testing the performance of web applications. *In Proceedings of the 3rd International Workshop on Software Quality Assurance*, pages 54-61, Portland, Oregon.

[15] Santos, I. S.; Santos Neto, P.; Moura, R. S.; Soares, A. C. B. (2010). Documentação Dirigida por Testes. In: *IX Simpósio Brasileiro de Qualidade de Software*, Belém, PA. Anais do IX Simpósio Brasileiro de Qualidade de Software, 2010. p. 25-40.